Object instance=each physical item that is being tracked by the system is represented by an instantiation of the appropriate object class.
OK=the user indicates he/she is done with this particular pop-up form.
SIGN=the user commits to the data he/she has entered.
Trigger=one of the independent variables in the equation that calculates the target attribute.

APPENDIX B

PSEUDO-CODE

```
Event Loop (Graph Mgr)
  Case DEFAULT:
  In attribute table, look up the default proc for the
    current attribute.
  If the default proc exists, call it.
  Break (next event).
    Case NEW:
      In the attribute table, look up the current field's
        attribute type and edit rules.
      In the attribute type table, look up the current
        field's attribute type and that of all its parents,
        and get all the edit rules.
      Call all the edit rules found in order.
      If any one of the edit rules fails Break (next
        event).
  Else (all edits passed)
    In the attribute table, look up the calculation rules
      triggered by the current attribute.
    For each such calculation rule, call it and any cal-
      culation it triggers.
    Pass the NEW event on to the current dialog node.
  Break (next event).
  Case CROSS_EDIT:
  In the attribute table, look up the current field's cross-
    _edit rules.
  Call all the crossedit rules found in order.
  If any one of the crossedit rules fails Break (next
    event).
  Break (next event).
  Case OK:
  Case SIGN:
  Case MORE:
  If any of the previous edits failed (this takes care of
    case where forms manager sends a slew of cross_
    edit events followed by an ok, sign, or more event)
    Ignore this event
  Else
    Pass the event on to the dialog node.
  Break (next event).
  Default:
  Pass the event on to the dialog node.
  Break (next event).
```

What is claimed is:

1. A method performed by a computer with an input means, the method comprising the steps of:

(a) storing an attribute identifier, an attribute-specific procedure identifier and an attribute type identifier in an attribute table entry in an attribute table;

(b) storing the attribute type identifier, a type-specific procedure identifier and a parent type identifier in a first type table entry in a type table;

(c) storing the parent type identifier as a second attribute type identifier and a parent type-specific procedure identifier as a second typespecific procedure identifier in a second type table entry in the type table;

(d) executing an application program that obtains data entered by a user via the input means; and

(e) executing a separate program upon determining that the data entered by the user in step (d) is an attribute having the attribute identifier stored in the attribute table, the execution of the separate program including the steps of:

(e1) locating the attribute identifier in the attribute table;

(e2) determining the attribute type identifier stored in the attribute table entry in the attribute table that stores the attribute identifier located in step (e1);

(e3) determining the parent type identifier in the first type table entry that stores the attribute type identifier;

(e4) executing a parent type-specific procedure identified by the parent type-specific procedure identifier stored as the second type-specific procedure identifier in the second type table entry that stores the parent type identifier determined in step (e3) as the second attribute type identifier;

(e5) executing a type-specific procedure identified by the typespecific procedure identifier stored in the first type table entry that stores the attribute type identifier determined in step (e2); and

(e6) executing an attribute-specific procedure identified by the attribute-specific procedure identifier stored in the attribute table entry in the attribute table that stores the attibute identifier located in step (e1).

* * * * *